Docket No.: 003829.P005
Express Mail No.: EM014065355US

UNITED STATES PATENT APPLICATION

FOR

A METHOD AND APPARATUS TO BALANCE FLOW LOADS IN A MULTIPURPOSE NETWORKING DEVICE

Inventors:

Tomasz J. Goldman
Christian Paulsen

Prepared By:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Blvd., 7th Floor
Los Angeles, California 90025-1026
(310) 207-3800

A METHOD AND APPARATUS TO BALANCE FLOW LOADS IN A MULTIPURPOSE NETWORKING DEVICE

BACKGROUND

Field of the Invention

[0001] The invention relates to networking. More specifically, the invention relates to traffic control in a distributed network.

Background

[0002] Routers in a traditional distributed network typically perform a routing function for a plurality of clients. Each client may send packets to be routed out over the internet, for example. A routing function is primarily a passthrough function and is dependent on egress bandwidth When more packets come in than the router is able to send out during a particular time window, the packets are queued at the output of the router to wait for an available sending opportunity. If a backup in the pipe is sufficient that the queue reaches a predetermined threshold, the router institutes a drop policy to drop packets at the output interface when the queue is too full. Among the known drop policies are: Dynamic Window, Slow Start, and Nagles Algorithm. While these policies work satisfactorily, where, as in this example, the only resource at issue is output bandwidth, they are not satisfactory for allocation of other scarce resources, such as, Central Processing Unit (CPU) time in a multipurpose networking device.

[0003] A multipurpose network device such as, for example, a combination router and file server performs, Transmission Control Protocol/Internet Protocol (TCP/IP) routing functions, file server functions, management functions, and other

activities based on incoming packets. These various activities must compete for shared resources within the device. As noted above, among the scarce shared resources are CPU time. While traditional systems allocate CPU time by task prioritization, task prioritization does not consistently provide desired and predictable performance.

BRIEF DESCRIPTION OF THE DRAWINGS

[0004] The invention is illustrated by way of example and not by way of limitation in the figures of the accompanying drawings in which like references indicate similar elements. It should be noted that references to "an" or "one" embodiment in this disclosure are not necessarily to the same embodiment, and such references mean at least one.

[0005] **Figure 1a** is a block diagram of the system of one embodiment of the invention.

[0006] **Figure 1b** is a bar diagram reflecting an example of load balancing in one embodiment of the invention.

[0007] **Figure 2** is a flow diagram of operation of flow control in one embodiment of the invention.

[0008] **Figure 3** is a flow diagram of packet handling in one embodiment of the invention.

DETAILED DESCRIPTION

[0009] While the discussion below is primarily in the context of balancing the load imposed on a processor by various data flows, the discussion can be generalized to any scarce resource. Such generalization is within the scope and contemplation of the invention.

[0010] **Figure 1a** is a block diagram of the system of one embodiment of the invention. A multipurpose network element 110 is coupled via a local area network (LAN) 102 to a plurality of clients 100. Clients 100 send both passthrough flows such as a flow including traffic stream 106 for which network element 110 may perform routing functions and flows to be handled internally by the element 110, such as one including traffic stream 104. These may include control flows, file server flows, etc. In one embodiment, typical traffic is in one form of Transmission Control Protocol/Internet Protocol (TCP/IP) packets.

[0011] Multipurpose device 110 includes a processor 114 coupled to a memory 116. Processor 114 handles the processing required for the various types of flows between the device and the outside world. The memory 116, may be for example, a read only memory (ROM). In one embodiment, the memory 116 stores one or more drop buffers 118 to implement a drop policy when the processor 114 is over utilized. The device 110 also includes an input interface 112 through which all flows from the client 100 to the device 110 flow. An output interface including an output queue 120 is provided for passthrough flows, such as, flow 106 to pass on to a distributed

network, such as internet 122. Any number of other devices may be coupled to internet 122 including, for example, an internet server 124.

[0012] As used herein, a "flow" is deemed to be an aggregation of packets of a particular type or category regardless of source. Thus, if two different clients were sending packets directed to internet 122. The packets from both clients would constitute one passthrough flow. It is desirable that all flows have some guaranteed Central Processing Unit (CPU) time to avoid starvation of the associated activity. Different types of packets will have different costs in terms of processor time required to service them. The packet load on the system depends on a number of factors. As noted above, one major distinction is between pass through flows (packets received by the network element 110) at one interface and passed through to another interface of network element 110 and flows to be handled internal to network element 110. Packets to be handled internally can further be differentiated between i) control and management packets used by a remote operator who monitors the system; and ii) packets to a file server within network element 110. Accordingly, a non-exclusive list of packets expected in one embodiment includes: i) packets routed through the network element without encryption; ii) packets routed through the network element with encryption; iii) management and control packets; and iv) packets addressed to the file server.

[0013] In one embodiment, the cost is used as a scaling factor to normalized the load of a flow or the processor 114. The system load is given by the equation $\vec{L} = (L_1, L_2, \ldots L_N)$ where there are N flows, and N is an arbitrarily large number. The load in packets per second (pps) for each flow F_i is given by the equation $L_i = C_i \times I_i$ where I_i is

the input rate in pps and C_i is the cost scaling factor. Thus, L_i is express in normalized pps.

[0014] To avoid over utilization of the scarce resource (here CPU time) $\Sigma_i L_i = L \leq P$ where P is the threshold processing rate measured in normalized pps. An ideal steady state for the system is given by the load vector $\vec{P} = (P_1, P_2, \ldots P_N)$ such that $\Sigma_i P_i = P$. The processor is over utilized when $L > P$ and a particular flow is excessive when $L_i > P_i$. A particular flow is under utilized if $L_i < P_i$. When a flow is under utilized, a portion of its share may be allocated to an excessive flow such that as long as $P \geq L$ all packets will be serviced. This share that may be reallocated is computed as a sum of flows over-utilizing their shares and unused capacity in flows under-utilizing their shares and is given by $\sum_{L_i > P_i} P_i + \sum_{L_i \leq P_i} (P_i - L_i) = S$. When the processor is over utilized, i.e., $L > P$, the excessive flow is scaled to bring $L \leq P$. An appropriate scale factor is given by $\frac{S}{\sum_{L_i > P_i} L_i} = K$.

[0015] **Figure 1b** is a bar diagram reflecting an example of load balancing in one embodiment of the invention. In this diagram, four flows are shown. Flow 1 has an allocated maximum steady state of ten packets, flow 2 has five packets, flow 3 has three packets and flow 4 has ten packets. Thus, P for this system is twenty-five. The load for flow 1 is six packets, for flow 2 is five packets, for flow 3 is fourteen packets and for flow 4 is six packets. Thus, the aggregate load L is thirty-one. This reflects an overloaded condition. Looking to the individual flows, flow 1 has an unused capacity of 4 packets, flow 3 has excessive usage of 6 packets and flow 4 has an over usage of 4 packets. The appropriate scaling factor calculated using the equation set

forth above is [(8 + 2) + 4]/(14 + 6) = 0.7. This indicates that flow 3 should be scaled down to ten packets and flow 4 should be scaled down to four packets. The numbers are arrived at using the scaling factor and an integer function, e.g., int (14 × 0.7) = int (9.8) = 10. Alternatively, a floor function could be used to make absolutely certain that the scaled load does not result in an overloaded condition. For example, floor (14 × 0.7) = floor (9.8) = 9. The integer function rounds to nearest integer while the floor function rounds to a next smaller integer.

[0016] To implement this scaling, a drop policy could be employed in which four consecutive packets from flow 3 and two packets from flow 4 are dropped. However, it is desirable to employ a random or pseudorandom (or simulated random) drop policy to avoid synchronization of multiple sources. A drop schedule is a function of K and the current load L_i. Thus, the reduction factor is given by $R_i = L_i \times (1 - K)$. In one embodiment, the drop factor may be implemented as a cylic buffer such as, drop buffer 118, in which a "1" indicates a packet drop and a 0 indicates the packet is serviced. After the drop or service decision is made, a drop schedule pointer is advanced to the next location in the drop buffer 118. In one embodiment, all drop schedules are based on a number of packets dropped out of eight possible packets. Thus, eight separate drop buffers may be retained in memory 116 and an appropriate one for the selected scaling factor is selected from the set of existing drop buffers. Packets to be dropped are dropped at the input interface 112.

[0017] **Figure 2** is a flow diagram of operation of flow control in one embodiment of the invention. At functional block 202, incoming packets are classified into flows. Classification may take the form of grouping packets based on the type of activity to

which they are directed, e.g., file serving or internet traffic. At functional block 204, a load of each flow for a current time slice is calculated. In one embodiment, the cost is merely based on a type of incoming packet. In another embodiment, cost may be calculated taking packet length into consideration. In such an embodiment, $C_i = C_o + f_o(l)$ where C_o is the minimum cost for a packet of the particular type and f_o is the function relating packet length to scarce resource usage.

[0018] At decision block 206, a determination is made if the aggregate load of all of the flows and exceeds a predicted steady state threshold. At decision block 208, if the aggregate flow does not exceed the steady state threshold, the determination is made if the processor is over utilized. If the processor is not over utilized, the system advances to the next time slice and no drop policy is employed at functional block 210. In one embodiment, a time slice is 200ms, other embodiment may employ longer or shorter time slices. If the load is greater than the steady state threshold at decision block 206, at decision block 214 a determination is made if the processor is under utilized. If the processor is under utilized the steady state threshold is raised to more efficiently use the processor. In one embodiment, there is a range in which the steady state threshold may be established. In one such embodiment, the rate at which the steady state threshold is reduced responsive to over utilization exceeds the rate at which the steady state threshold is increased responsive to under utilization. In another embodiment, the steady state threshold is fixed and unchangeable after manufacture. In such an embodiment, blocks 212-216 are effectively omitted.

[0019] After the steady state threshold is lowered or if at decision block 214 the processor is not under utilized, or after the steady state threshold is raised, a drop policy for the next time slice is calculated at functional block 216. Then at functional block 220, the processor selects an appropriate drop buffer reflecting the drop factor calculated at functional block 218. At functional block 222, the system advances at the next time slice and applies the drop policy reflected in the previously selected drop buffer. Accordingly, the drop policy for a time slice T_1 is established based on actual traffic in time slice T_o.

[0020] **Figure 3** is a flow diagram of packet handling in one embodiment of the invention. At functional block 302, a packet is received at the input interface. At functional block 304, the packet is classified into a flow. At functional block 306, the cost of the packet is calculated and added to the cost for the corresponding flow during the current time slice. The determination is made at decision block 308 whether the flow was excessive during the prior time slice. If the flow was excessive, a determination is made whether the drop buffer indicates that the packet should be dropped at functional block 310. If the drop buffer indicates the packet should be dropped, the packet is dropped at the input interface at functional block 312. If the flow was not excessive in the prior time slice, or the drop buffer does not indicate the packet is to be dropped, the packet is serviced at functional block 314.